SMU BIA

Co-Learning Session

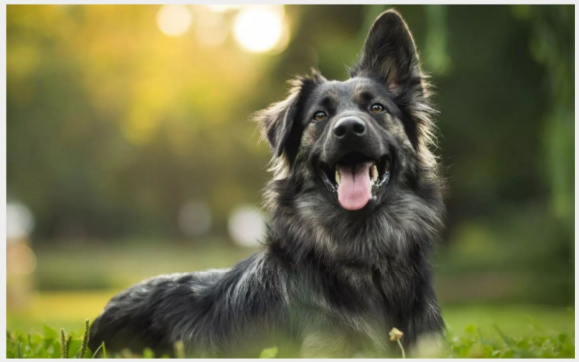
Week 3: Classification

Modified from the 9th DAP. Made by YZ, updated by Larry for the 11th.

Classification

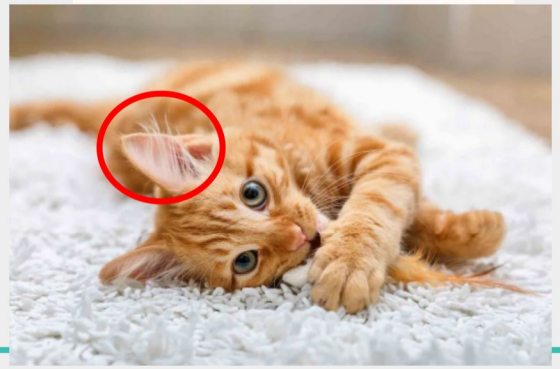
These are cats:

These are dogs:

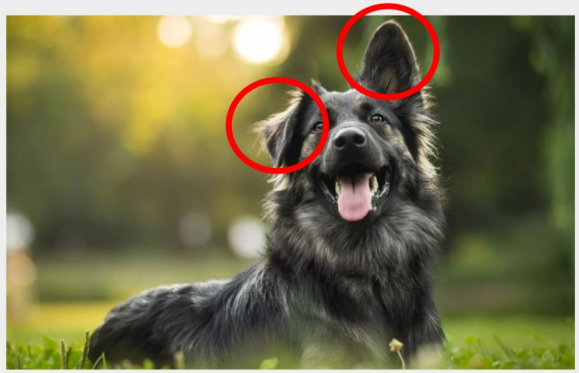


Classification

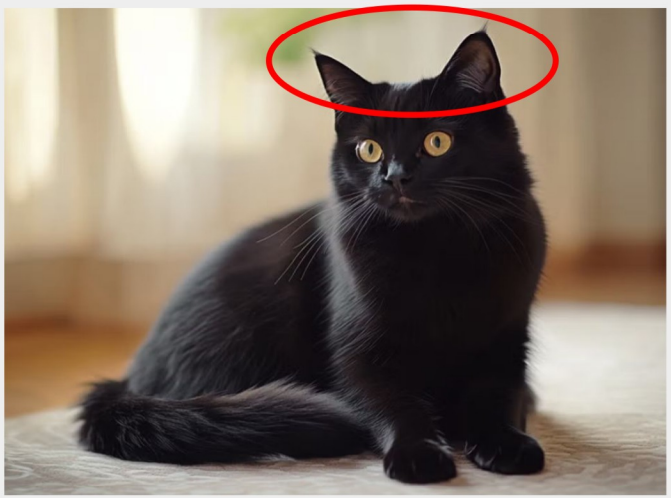
These are cats:



These are dogs:



Classification



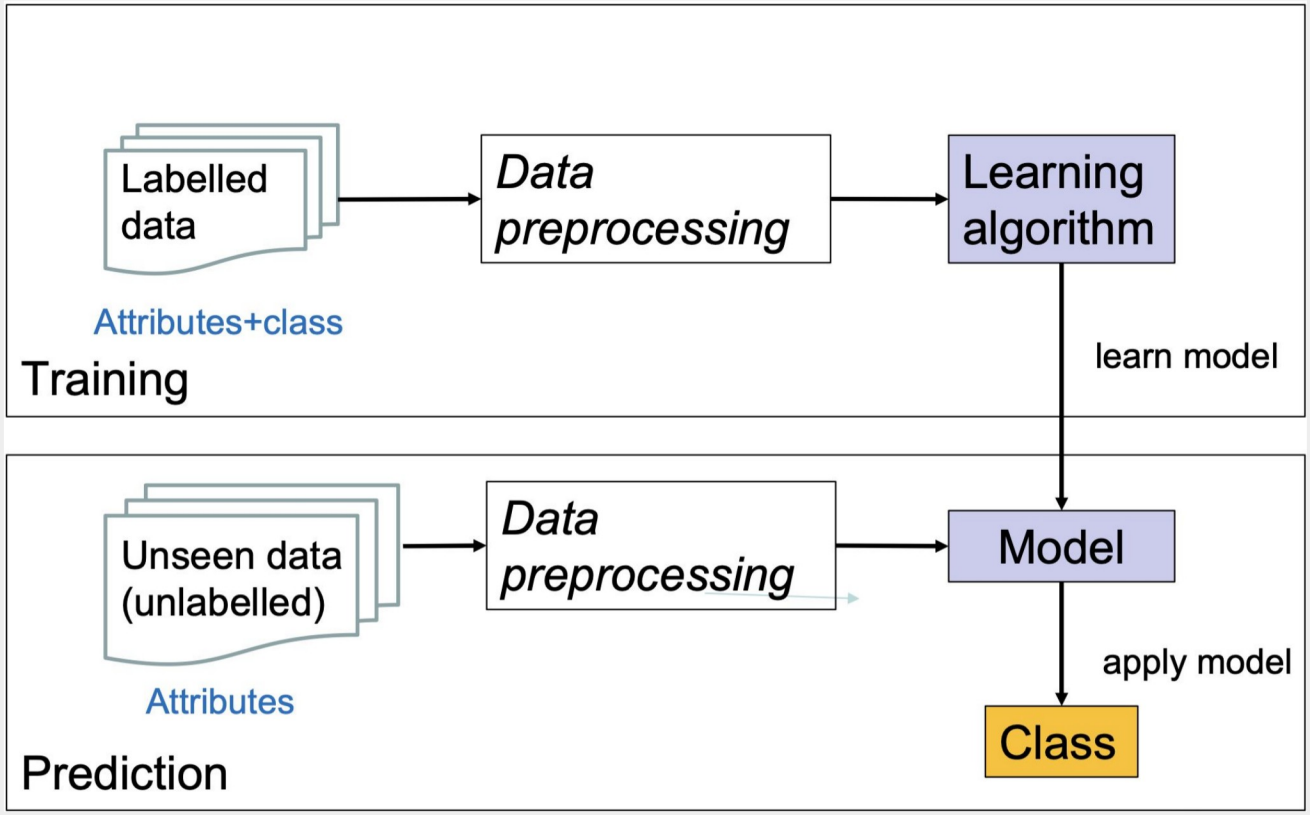
Cat

What is Classification?

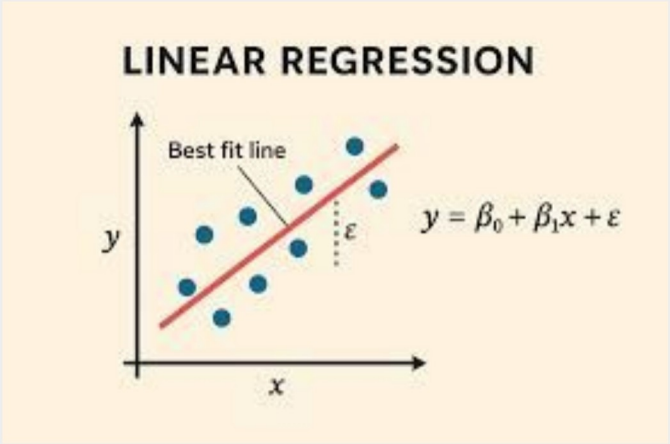
- A classification problem is a type of **supervised learning** task where the goal is to predict which **category or class** an item belongs to as accurately as possible.
- Classification assigns a label from a **small, finite set of possible values.**

A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model, and test set used to evaluate its accuracy

Classification illustrated



Remember Linear Regression?

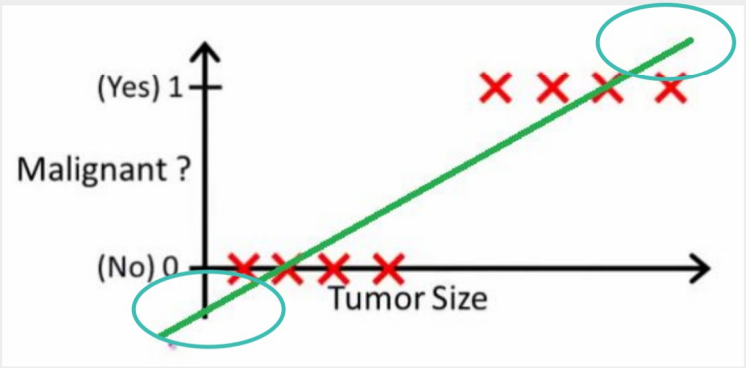


Can we use it for Classification too?

Problem #1: Predicted value is continuous, not probabilistic

Binary classification goal: predict the probability of class 1.

- Probabilities always lie between **0 and 1**.
- 0 → event very unlikely, 1 → event certain.



Linear regression problem:

- Predicts a number on a **continuous scale** ($-\infty$ to $+\infty$).
  This number is **not constrained**, so it can go below 0 or above 1.

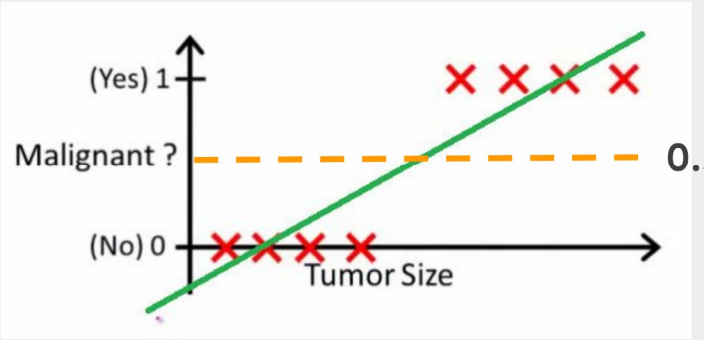
Why this matters:

- Values outside [0,1] cannot be interpreted as probabilities for discrete categories.
- Makes linear regression **inaccurate and unreliable** for classification.

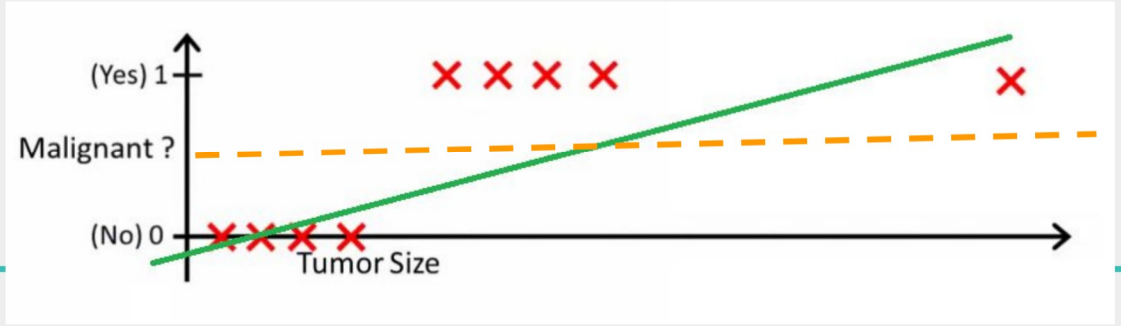
Problem #2: Sensitive to imbalanced data

Linear regression tries to **fit all points numerically** to reduce squared error.

a single extreme point can tilt the best fit line, reducing accuracy and increasing the chance for a biased model



Finds the new best fit line

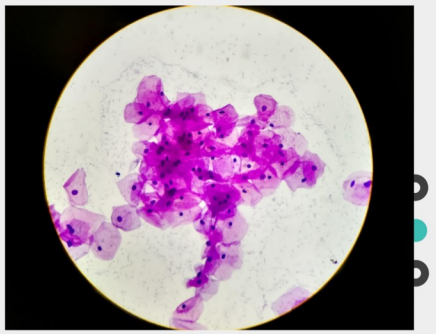


Lab session!

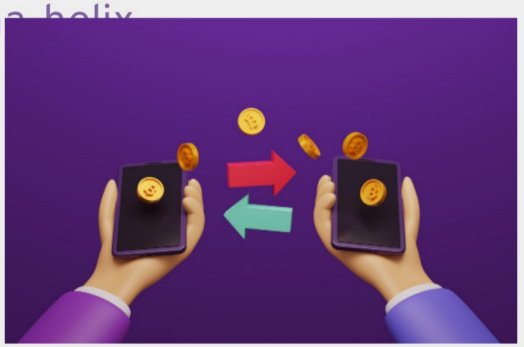
Let's try to experiment with outliers.

https://claude.ai/public/artifacts/2607f725-337d-45d9-9081-b60ea9437057

Examples of classification tasks



- Predicting Tumor Cells As Benign Or Malignant
- Classifying Credit Card Transactions as legitimate or fraudulent
- Classifying Secondary Structures of protein as alpha-helix, beta-sheet, or random coil in biology



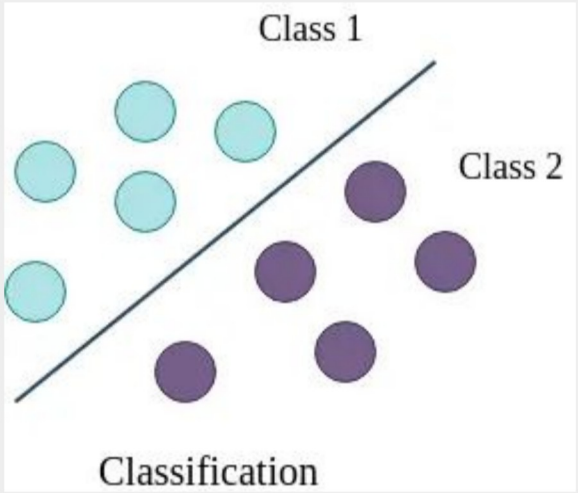
- Categorising News Stories As Finance, weather, entertainment, sports, etc

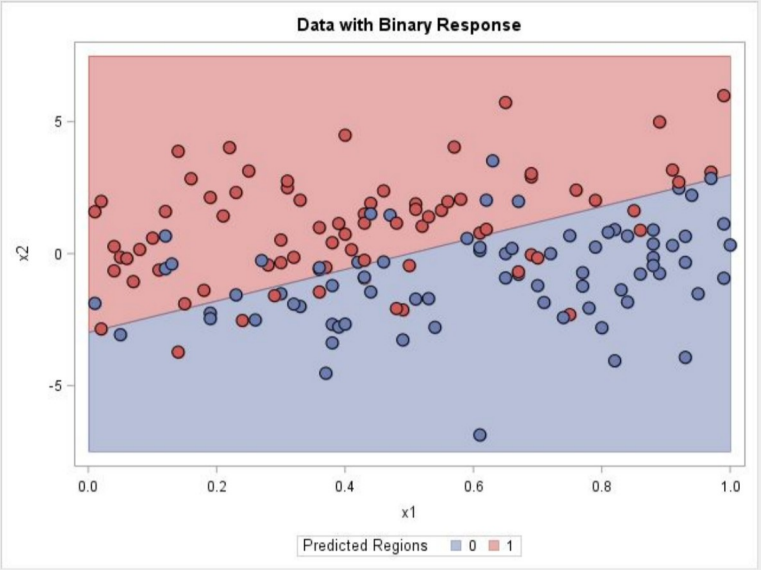
Types of classification

- Binary Classification
- Multi-Class Classification
- Multi-Label Classification

Binary Classification

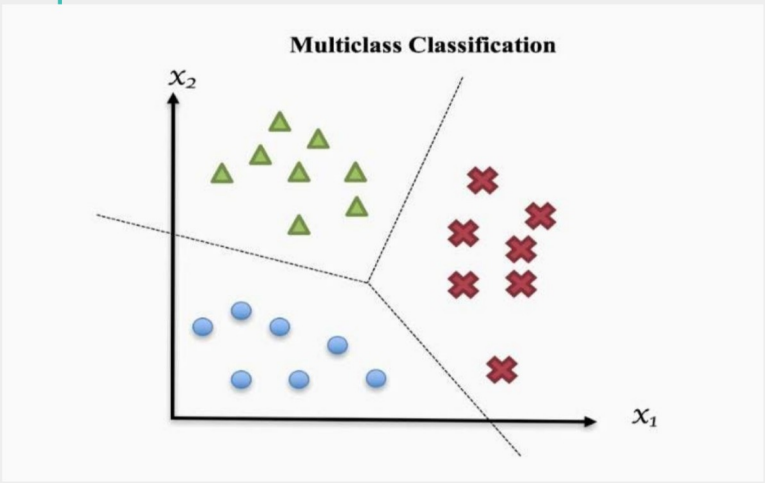
Binary Classification: classification tasks that have **two class labels**

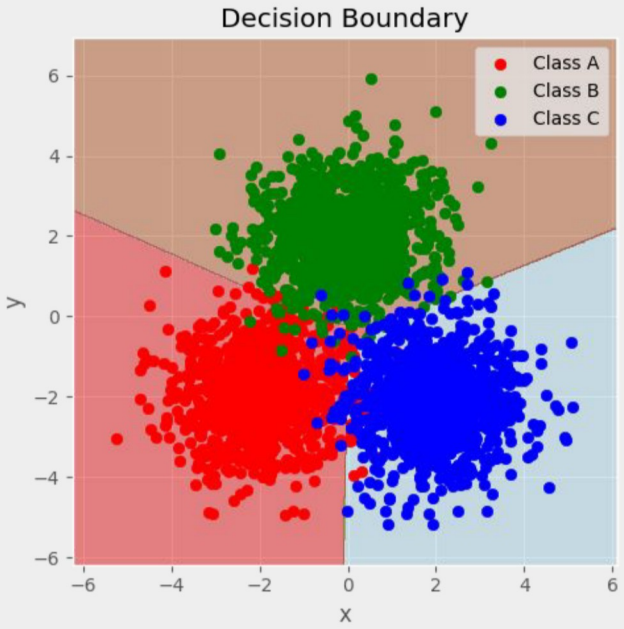




Multi-class Classification

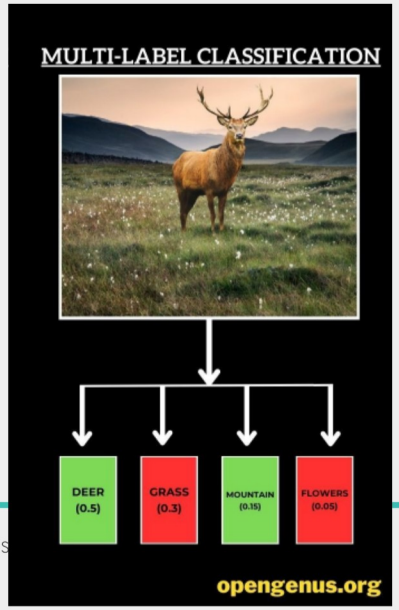
Multi-Class Classification: classification tasks that have **more than two class labels**.

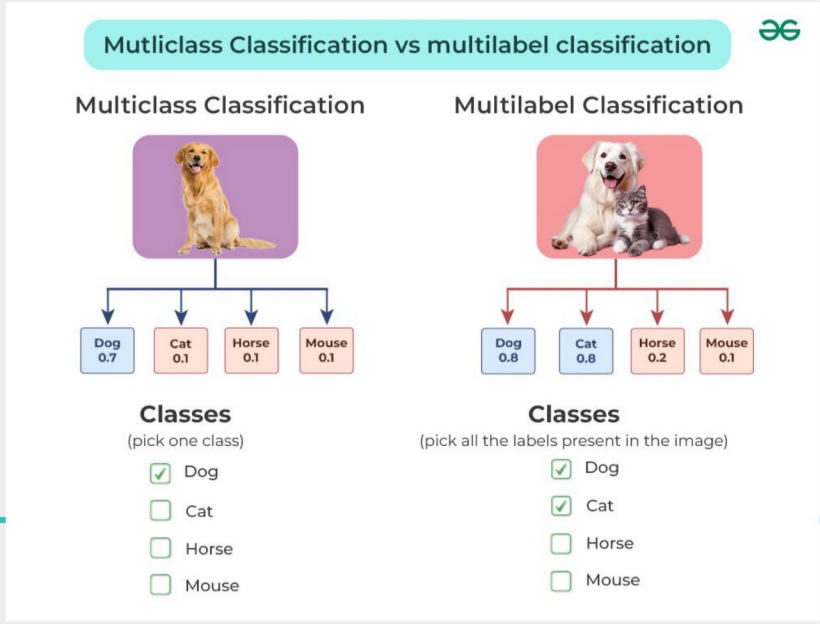




Multi-Label Classification

Multi-Label Classification: classification tasks that have **two or more class labels**, where one or more class labels may be predicted for each example.





Logistic Regression & Probabilistic Binary Classification

Things covered:

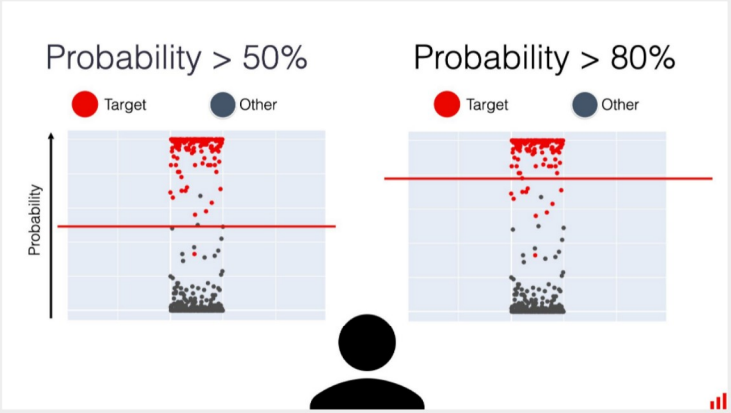
- Binary Classification in practice
- Logistic Regression Mechanics
- Odds, Log-Odds and Probability
- Surprise, Entropy and Cross-Entropy

Classification & Probabilities

Goal:

To estimate the value of $P(Y = 1 \mid X)$

- Label comes from thresholding



Differences between Types

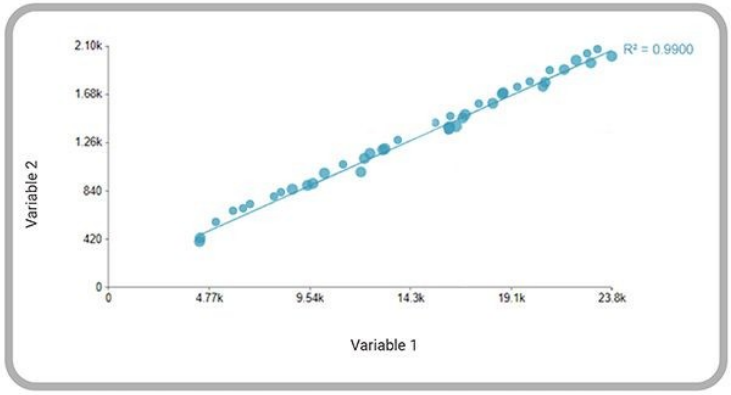
- **Binary Classification:**
  Limited to one probability, and classifies between either or. For eg. Whether an email is spam or not spam

- **Multiclass Classification:**
  Classifies between one of K objects. For eg. Classifying a hand drawn digit from 0-9

- **Multilabel Classification:**
  K number of yes/ no outputs. For eg. Classifying an image with tags (Adventure, Sports, etc.)

Where Linear Regression Fails

- Outputs of Linear Regression: ANY real number
- What classification specifically requires: Valid probability



Linear Regression Graph: How Can It Make Sense of Your Data?

Logistic Regression: What it predicts

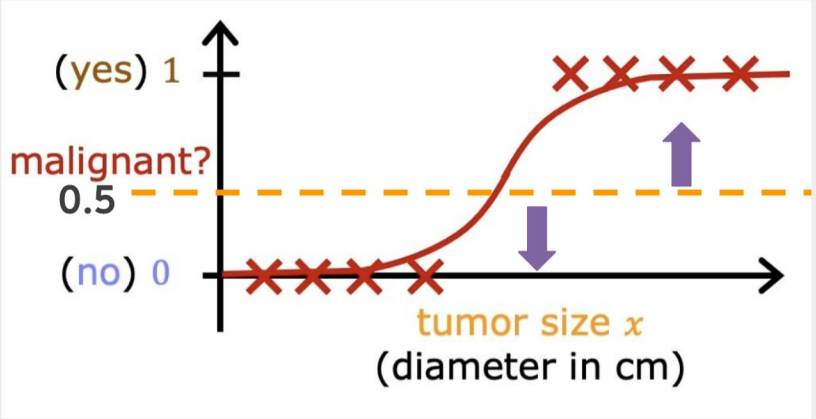
$$\hat{y} = P(Y = 1 \mid X)$$

- Confidence can be interpreted in a human way

- Easier decision making

Decision Boundary

- **Purpose:** To turn **continuous probability predictions** into **discrete class labels**, because classification outputs must be a label, not just a number
- Non-linear classifiers → curved or flexible boundary
- Most common threshold = 0.5 → probability ≥ 0.5 → Class 1, probability < 0.5 → Class 0



LAB: Trying out different boundaries

https://claude.ai/public/artifacts/b64545e3-0168-42d4-a9c6-ccb059427277

The Linear Score

$$z = w^\top x + b$$

w: Feature weights (transpose)

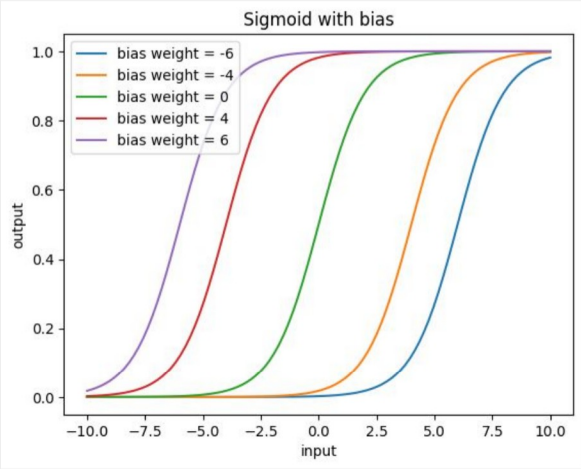
x: Input features

b: Bias

z: Linear score

What is Bias?

- Gives a baseline log-odds
- Prediction when all other features are zero
- Encodes prior tendency



Worked Example: Loan Approval Model

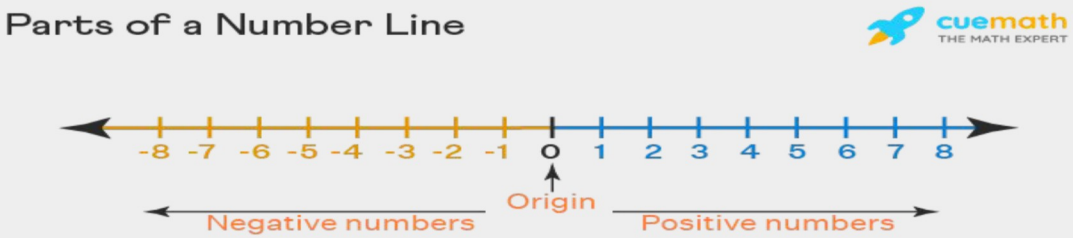
- Loan Approval Model

$$z = 0.5(\text{Income}) - 0.9(\text{Debt}) - 1$$

- Income increases approval
- Debt is Penalised
- Bias Favours rejections

The Problem with Z

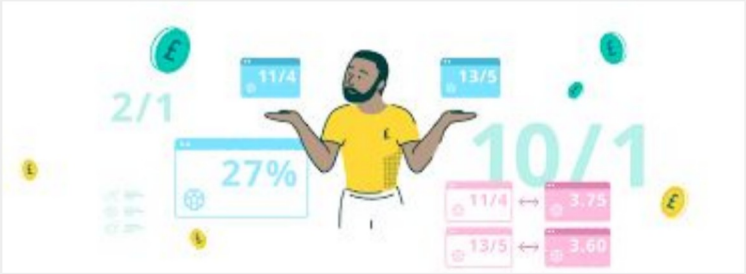
- Z is not a probability, $z \in (-\infty, +\infty)$
- Probability has to be bounded from (0, 1)



Odds

$$\text{odds} = \frac{p}{1-p}$$

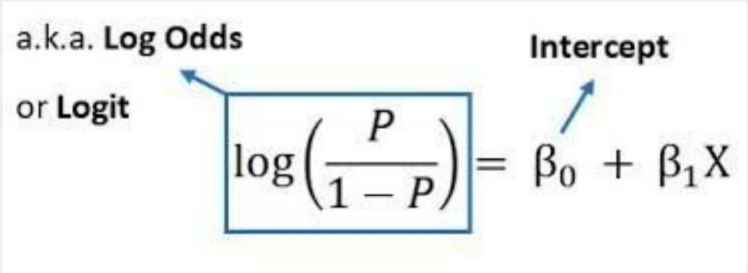
- Compares success vs failure
- Range 0 → ∞



Logits: Log-Odds

$$z = \log \frac{p}{1-p}$$

- Maps probabilities to real numbers
-Logistic regression models this linearly



Mini-Lab: Scaling Log-Odds

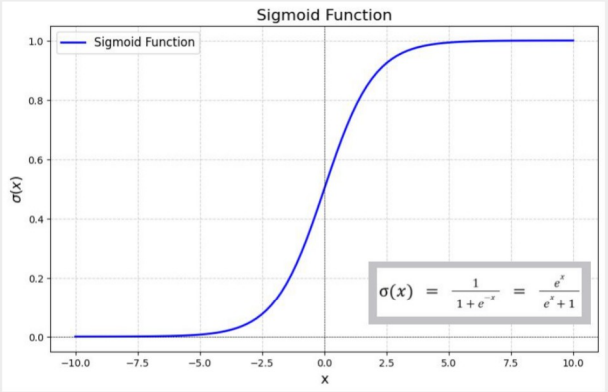
https://claude.ai/public/artifacts/ce0a7b13-bd72-411b-aa97-025322aea689

Getting the Sigmoid Function

Sigmoid Function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

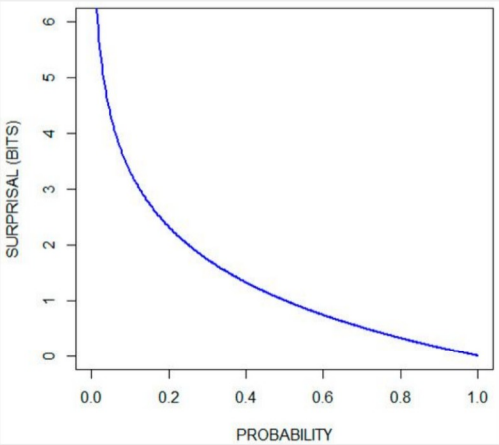
- Inverse of Log-odds
- Smooth, much more easily differentiable



Surprise

$$\text{Surprise}(y) = -\log P(y)$$

- Rarer the event -> Higher surprise
- Confident wrong predictions hurt more

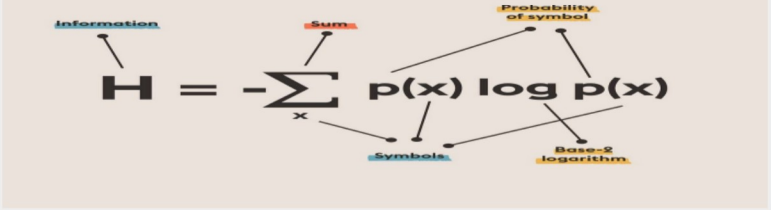


Mini-Lab: Scaling Surprise & Probability

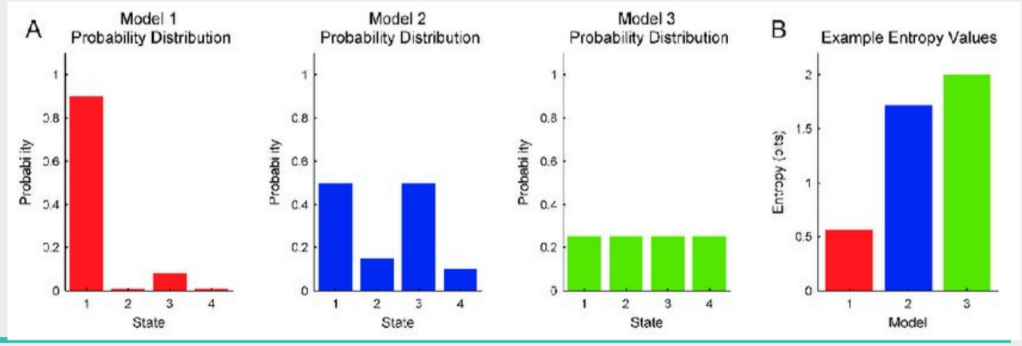
https://claude.ai/public/artifacts/17ea8b4e-5b33-477b-aec9-bd194cce6650

Entropy

$$H(Y) = \mathbb{E}[-\log P(Y)]$$



- Expected/ Average Surprise
- Measures uncertainty



Examples of entropy

- Fair Die → Highest Entropy

Probability: 0.1667

Entropy: 1.79

- Biased Die → Entropy is lower

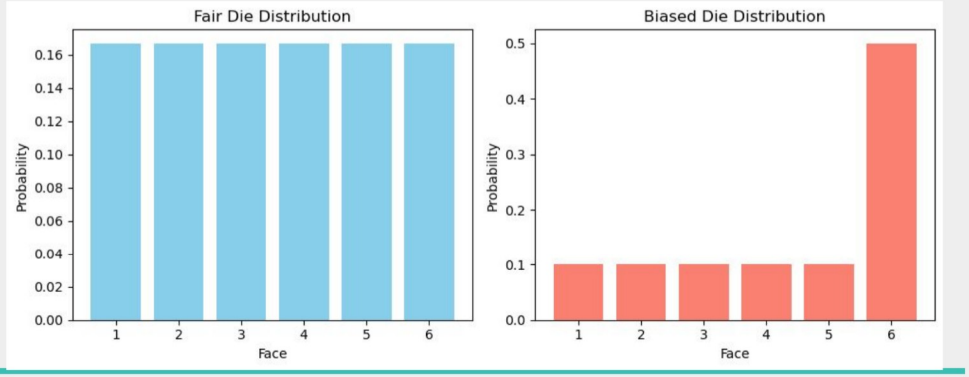
Probability: 0.5/ 0.1

Entropy: 1.50

- Unfair Die → Zero Entropy

Probability: 1.0/ 0.0

Entropy: 0



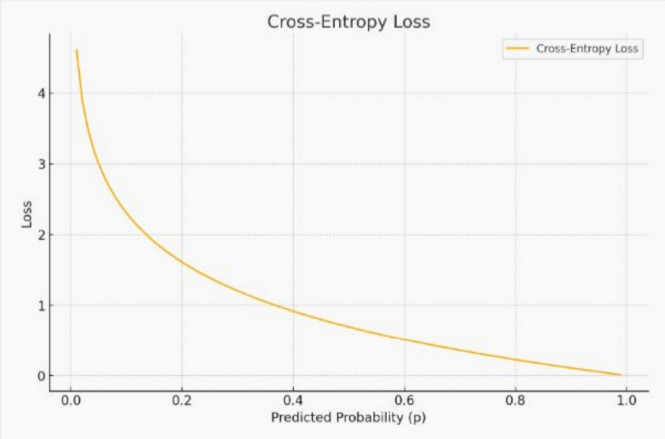
Cross Entropy

$$H(p, q) = -\sum p(y) \log q(y)$$

- True Distribution vs Model Prediction
- Measures Model Prediction
- Theoretical

Binary Cross Entropy Loss

$$\mathcal{L} = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$$



Why Cross Entropy Works

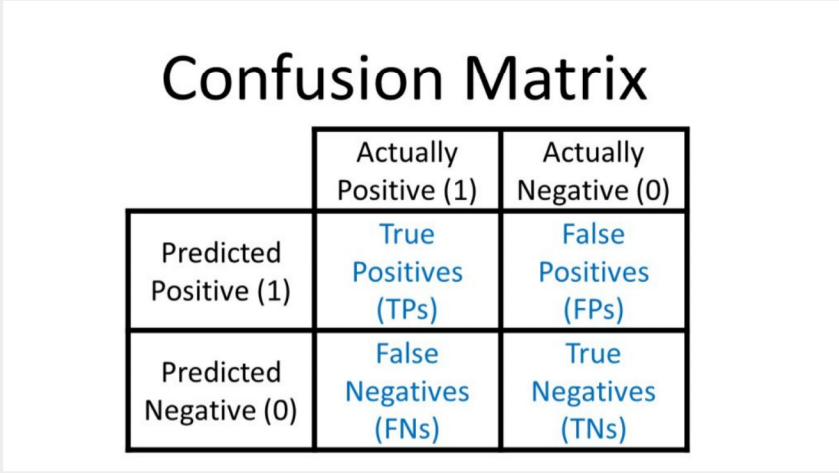
$$\frac{\partial \text{Loss}}{\partial \text{prediction}} = \frac{\text{prediction} - \text{truth}}{\text{prediction}(1 - \text{prediction})}$$

- Confident & wrong → Large Gradient Change
- Confident & Correct → Small Gradient Change

2.3 Evaluation metrics

The Confusion Matrix

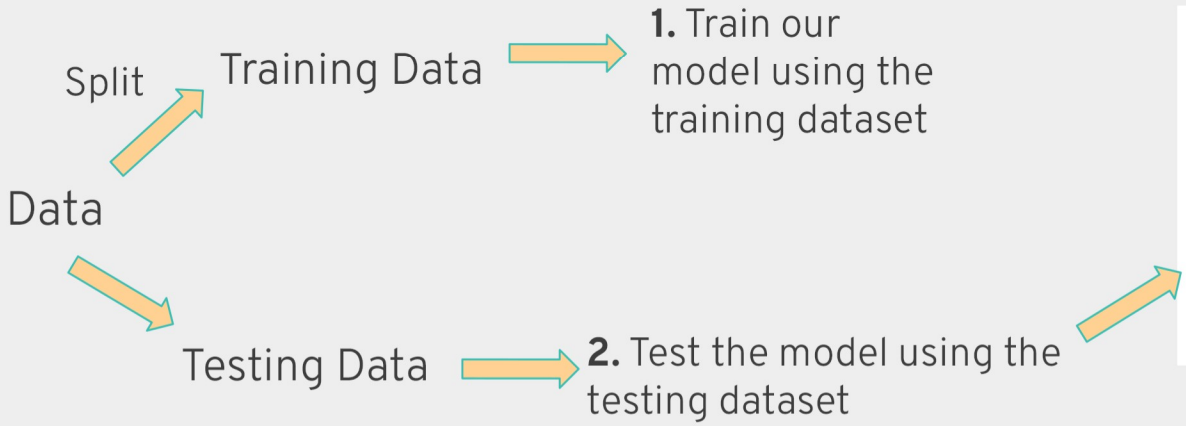
- A simple table used to measure **how well** a classification model is performing. It compares the predictions made by the model with the actual results
- This helps you understand where the model is making mistakes so you can improve it. It breaks down the predictions into four categories



4 Types of predictions

- **True Positive (TP):** The model **correctly** predicted a **positive** outcome
- **True Negative (TN):** The model **correctly** predicted a **negative** outcome
- **False Positive (FP):** The model **incorrectly** predicted a **positive** outcome i.e the actual outcome was negative. It is also known as a Type I error.
- **False Negative (FN):** The model **incorrectly** predicted a **negative** outcome i.e the actual outcome was positive. It is also known as a Type II error.

Example: Spam vs Non-spam



3. Confusion matrix

		Predicted: Spam	Predicted: Non-spam
Actual	Spam	600 (True positive)	300 (False negative)
Actual	Non-spam	100 (False positive)	9000 (True negative)

Metrics based on Confusion Matrix Data

1. Accuracy
2. Precision
3. Recall
4. F1 Score
5. F-Beta Score

Accuracy

Accuracy shows how many predictions the **model got right out of all the predictions.**

Problem: It gives idea of overall performance but it can be **misleading** when one class is more dominant over the other.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision

"When it predicts positive, how often is it correct?"
Precision is a good measure to use when the costs of False Positive is high and False Negative is low. e.g. email spam detection.
→ Minimise False Positives

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall

"When it's actually positive, how often does it predict positive?"

Recall is a good measure to use when the costs of False Negative is high. e.g. fraud detection, cancer detection.

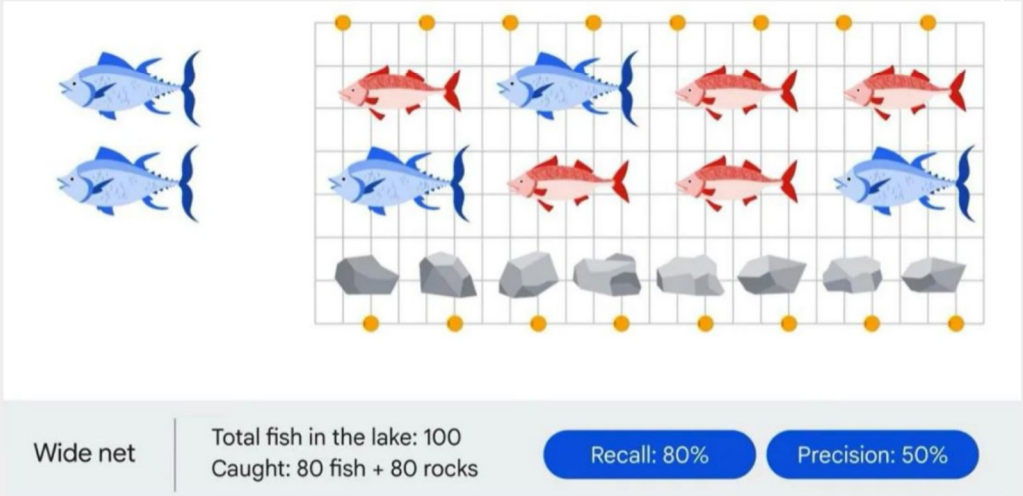
$$\text{Recall} = \frac{TP}{TP + FN}$$

Analogy for Recall and Precision

True Positives: catching the fish (80)
False Positives: Catching the rocks (80)
False negatives: uncaught fish (20)

Recall = 80 / (80 + 20) = 0.8

Precision = 80 / (80 + 80) = 0.5

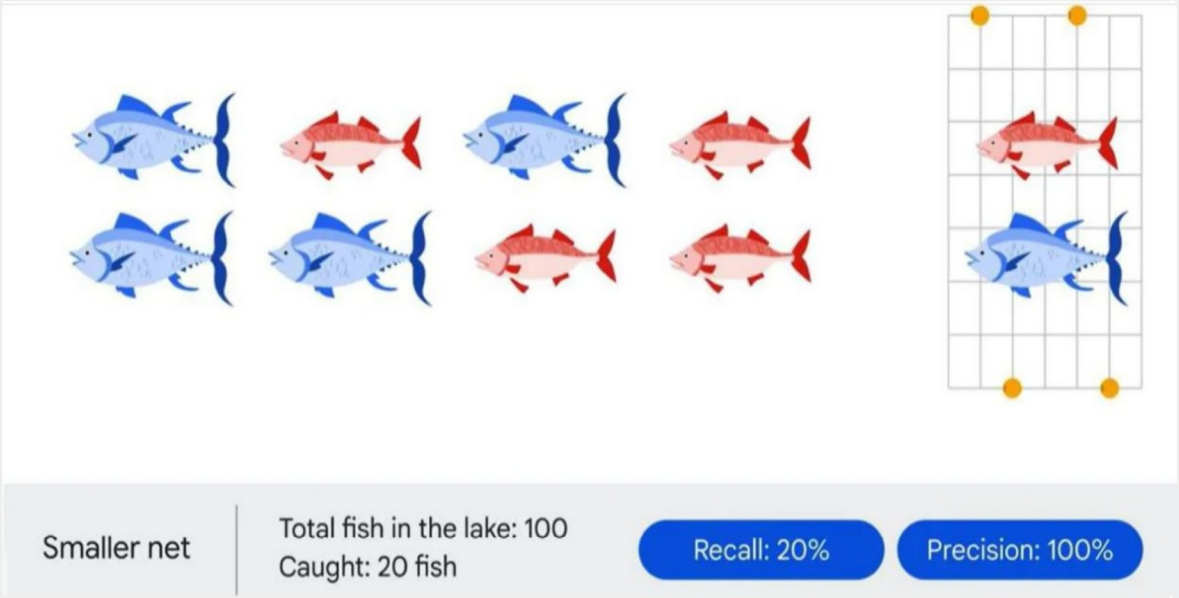


Analogy for Recall and Precision

TP= 20 fish
FP= 0
FN = 80 fish

Recall: 20 / (20 + 80) = 0.2

Precision: 20 / 20 + 0 = 1



Precision and Recall

Recall: Correct Positive Guesses / Total Positive Labels = **TP/TP+FN**
(focuses on False Negatives)

Precision: Correct Positive Guesses / Total Positive Guesses = **TP/TP+FP**
(focuses on False Positives)

	Actual Negative	Actual Positive
Predicted Negative	True Negative (TN)	False Negative (FN)
Predicted Positive	False Positive (FP)	True Positive (TP)

F1 Score

Balanced F-score or F-measure or F1

- Harmonic mean of precision and recall
- Defined as

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

- F-score provides a measure of accuracy without bias toward precision or recall
- The higher the F-score, the better the classification model

F-Beta Score

The Fβ-score is a generalised version of F1-score that allows you to give **more importance to either precision or recall**.

- **Precision**: The proportion of predicted positives that are actually correct.
- **Recall**: The proportion of actual positives that the model correctly identified.
- **β (beta)**: A positive real number that determines the weight of recall in the score.

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}}$$

How β (beta) affects the score

- **β = 1**: Precision and recall are equally important. This becomes the F1 Score.
- **β > 1**: More weight is given to recall. Useful when false negatives are more costly.
- **β < 1**: More weight is given to precision. Useful when false positives are more costly.

https://claude.ai/public/artifacts/c1ac64b6-e97d-4582-8587-eb4e7f2a24ef

Lab session!

Let's explore using the different metrics:

https://claude.ai/public/artifacts/53c64ed2-774f-477d-9a3c-f6dd1897d2d0

Summary/quiz!

Accuracy:	Measures how many predictions were correct overall
Precision:	Measures how many predicted positives were actually correct
Recall:	Measures how many actual positives were correctly identified
F1 Score:	Measures the balance between precision and recall
F-Beta Score:	Measures precision and recall while giving more importance to one over the other

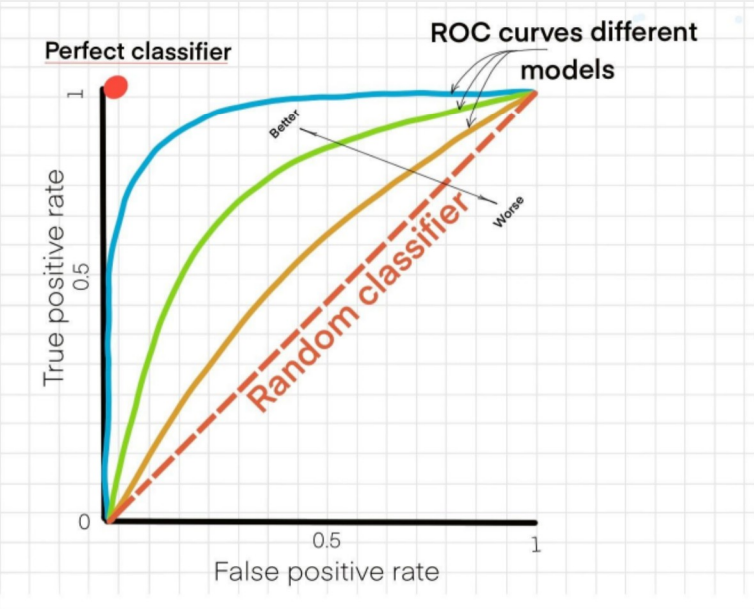
Model evaluation: ROC curve and AUC

Model Selection

We use the **Receiver Operating Characteristic (ROC) Curve**:

- True Positive Rate **(how often the model correctly predicts the positive cases)** on y-axis
- False Positive Rate (**how often the model incorrectly predicts a negative case as positive)** on x-axis

Graphical representation of the performance of a **binary classification** model at different thresholds

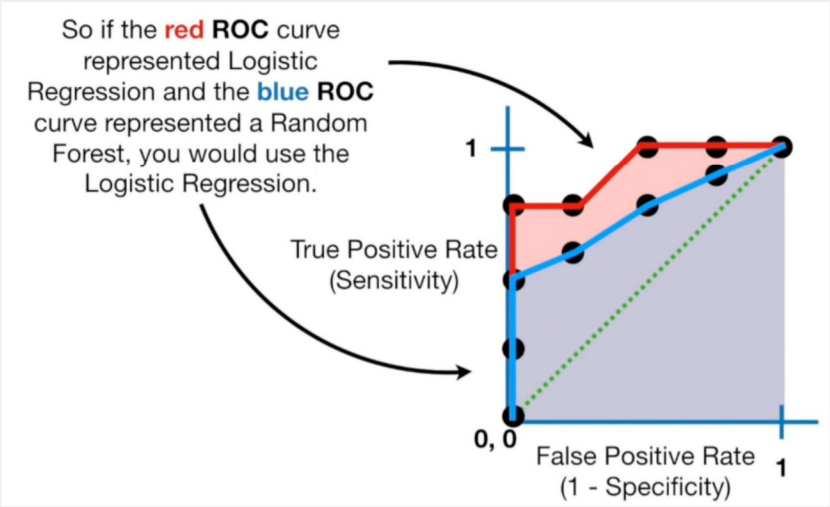


Model Selection

Area under the Receiving Operating Characteristics Curve (AUC):
Quantifies the **discriminative power** of a classification model (similar to R Squared)

- Measures overall ability to separate classes
- **Higher AUC = better discrimination**

An AUC of **0.73** means that **73% of the time**, the model gives a **higher score to a real positive case than to a real negative case.**
AUC = 0.5 is the same as randomly guessing



***Discrimination = the model's ability to distinguish between the two classes**

2.4 Multiclass / Multilabel Classification

The Multiclass Challenge

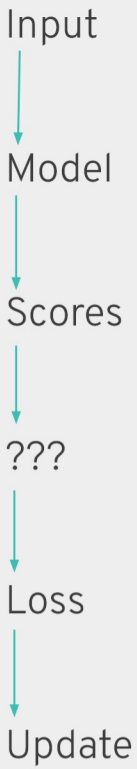
- **The Problem: Beyond Binary**
- **Binary SVMs work perfectly for 2 classes but real world problems need more:**
  - Email classification: Spam, Promotional, Personal….
  - Medical diagnosis: Healthy, Flu, COVID-19, Pneumonia
- **Why simple extensions fail:**
- ❌ **Single Hyperplane Approach:**
  - “Just find one boundary for all classes” → Problem: Mathematically impossible for N > 2 classes
- ❌ **Threshold-Based Method**
  - “Use multiple thresholds on one decision function” → Problem: Classes aren’t naturally ordered (What is between a cat and a dog)

One-vs-Rest & One-vs-One

Strategy 1: One-vs-Rest (OvR)	Strategy 2: One-vs-One (OvO)
• The Approach: • Train N binary classifiers (one for each class):	• The approach: • Train N(N-1)/2 binary classifiers (one for each pair)
• Classifier i: Class i vs All Other classes	• Classifier (i,j): Class i vs Class j only
• Example: {Cat, Dog, Bird} → 3 classifiers: • Cat vs. {Dog, Bird} • Dog vs. {Cat, Bird} • Bird vs. {Cat, Dog}	• Example: {Cat, Dog, Bird} → 3 classifiers: • Cat vs Dog • Cat vs Bird • Dog vs Bird

Cross Entropy Loss

How do we train multiclass classifiers?



- Evaluation tells us *how good* predictions are

- Training requires:
  - Probabilities
  - A loss that measures how wrong they are

Why Softmax + Cross Entropy

- Honest Confident Scores

- Strong penalty for confident wrong predictions

- Works for any number of categories

Softmax: Probabilities over multiple classes

$$\text{softmax}(z_k) = \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$$

- Converts arbitrary scores into probabilities
- All probabilities sum to 1
- Classes compete with each other

Multiclass Cross Entropy Loss

$$\mathcal{L} = -\sum_{k=1}^{K} y_k \log(\hat{y}_k)$$

- One hot true label vector [0, 0, 1, 0]
- Softmax probabilities [0.1, 0.2, 0.6, 0.1]
- Therefore -log(0.6)

2.5 Advanced Classification Methods

NB and KNN

Naive Bayes (NB)

- Supervised Learning Algorithm
- Predicts a class using **probability**
- Naively assumes all features (x_1,x_2, x_3 …)are **independent of each other**
- “Bayes" part of the name refers to Bayes’ Theorem.
- Applications
  - Spam email filtering
  - Text Classification
  - Credit Scoring
  - Recommendation System
  - … and more

Bayes Theorem

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$

P (y|X): Probability of class y given features

E.g. class of y could be “Yes”, “No”

P (X|y): Probability of features X given class y

P (y): Prior probability of class y

P (X): Marginal likelihood of combination of features

Dating App Prediction

Entry	Tall	Shared Interests	Compatible Horoscope	Match
1	Yes	Yes	Yes	**Yes**
2	Yes	Yes	No	**Yes**
3	Yes	No	Yes	**Yes**
4	Yes	Yes	Yes	**Yes**
5	No	Yes	Yes	**Yes**
6	No	No	Yes	**No**
7	Yes	No	No	**No**
8	No	Yes	No	**No**

Dating App Prediction

Next Contestant:

Tall = Yes
Shared Interests = Yes
Compatible Horoscope = No

Will Mr Bayes pop his balloon?

Naive Bayes

Entry	Tall	Shared Interests	Compatible Horoscope	Match
1	Yes	Yes	Yes	Yes
2	Yes	Yes	No	Yes
3	Yes	No	Yes	Yes
4	Yes	Yes	Yes	Yes
5	No	Yes	Yes	Yes
6	No	No	Yes	No
7	Yes	No	No	No
8	No	Yes	No	No
9	Yes	Yes	No	?

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$

P (y|X): Probability of class y given features X

P(Match = Yes | X) = ?

Or

P(Match = No | X) = ?

Naive Bayes

Entry	Tall	Shared Interests	Compatible Horoscope	Match
1	Yes	Yes	Yes	Yes
2	Yes	Yes	No	Yes
3	Yes	No	Yes	Yes
4	Yes	Yes	Yes	Yes
5	No	Yes	Yes	Yes
6	No	No	Yes	No
7	Yes	No	No	No
8	No	Yes	No	No
9	Yes	Yes	No	?

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$

P (y): Prior probability of class y

P(Match = Yes) =

P(Match = No) =

Naive Bayes

Entry	Tall	Shared Interests	Compatible Horoscope	Match
1	Yes	Yes	Yes	Yes
2	Yes	Yes	No	Yes
3	Yes	No	Yes	Yes
4	Yes	Yes	Yes	Yes
5	No	Yes	Yes	Yes
6	No	No	Yes	No
7	Yes	No	No	No
8	No	Yes	No	No
9	Yes	Yes	No	?

$$P(y|X) = \frac{P(X|y)\,P(y)}{P(X)}$$

P (X|y): Probability of features X given class y

P(X | Match = Yes)

P (Tall = Yes | Match = Yes) = 4 / 5
P (SI = Yes | Match = Yes) = 4 / 5
P (CH = No | Match = Yes) = 4 / 5

P(X | Match = No)

P (Tall = Yes | Match = No) =
P (SI = Yes | Match = No) =
P (CH = No | Match = No) =

Naive Bayes

Entry	Tall	Shared Interests	Compatible Horoscope	Match
1	Yes	Yes	Yes	Yes
2	Yes	Yes	No	Yes
3	Yes	No	Yes	Yes
4	Yes	Yes	Yes	Yes
5	No	Yes	Yes	Yes
6	No	No	Yes	No
7	Yes	No	No	No
8	No	Yes	No	No
9	Yes	Yes	No	?

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$

P (X) is a **constant**

- observed features X are fixed for a given data point

Naive Bayes

Entry	Tall	Shared Interests	Compatible Horoscope	Match
1	Yes	Yes	Yes	Yes
2	Yes	Yes	No	Yes
3	Yes	No	Yes	Yes
4	Yes	Yes	Yes	Yes
5	No	Yes	Yes	Yes
6	No	No	Yes	No
7	Yes	No	No	No
8	No	Yes	No	No
9	Yes	Yes	No	?

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$

P(Match = Yes | X)
= P(X | **Yes**) * P(**Yes**) / P(X)
= (⅘ * ⅘ * ⅘ * ⅝) / P(X)
= 0.32 / P(X)

P(Match = No | X)
= P(X | **No**) * P(**No**) / P(X)
= (⅔ * ⅔ * ⅓ * ⅜) / P(X)
= 0.056 / P(X)

Naive Bayes

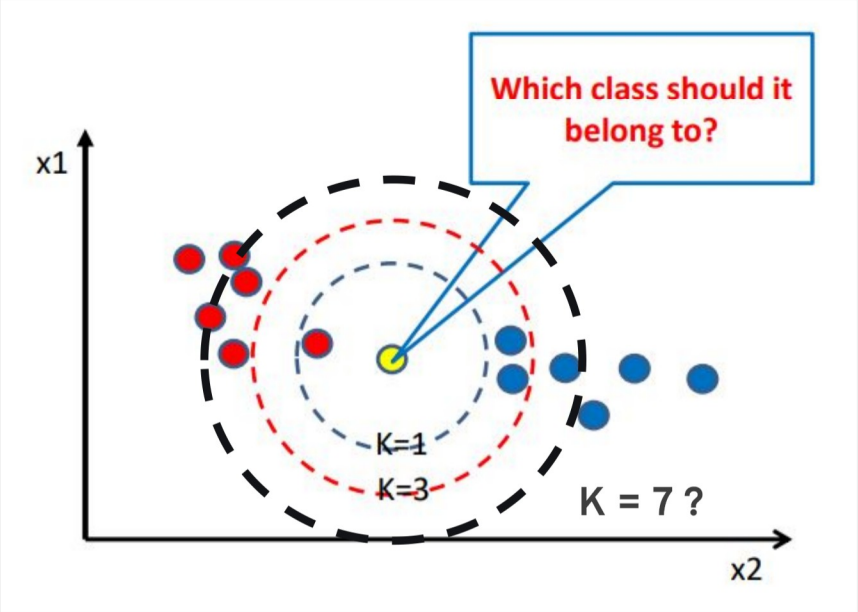
Pros	Cons
Simple probabilistic classifier Very few number of parameters	Assumes that features are independent • not always hold in real-world data
Fast Prediction	Can be influenced by irrelevant attributes
Effective with a large number of features	Poor generalization • May assign zero probability to unseen events
Performs well even with limited training data	

https://colab.research.google.com/drive/1STgOkczKMhEMgfBbrQYM3YzbrLdHOSc2#forceEdit=true&sandboxMode=true

K-Nearest Neighbours

- Supervised Learning Algorithm
- Finding the **"k"** closest data points (neighbors) to a given input
- Makes a prediction/ assigns a category based on the **majority class** (for classification)
- Lazy learner algorithm
  - does not learn from the training set immediately
  - performs computations only at the time of classification
- **Applications:**
  - Recommendation System
  - Spam Detection
  - Customer Segmentation
  - Speech Recognition

K-Nearest Neighbours



K-Nearest Neighbours

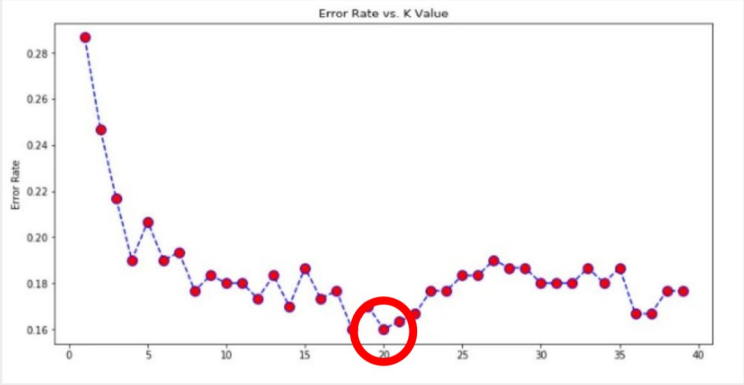
1. **Select the optimal value of K**
   - K → number of nearest neighbors
2. **Calculate distance between target point and training data points**
   - To measure the similarity between target and training data points
3. **Find the K Nearest Neighbors**
   - The k data points with the **smallest** distances to the target point are **nearest neighbors.**
4. **Vote for Classification** (Majority Voting)
   - Algorithm picks the category that appears the **most** from the K closest points

Selecting the Number of K

- If k is **too small**
  - sensitive to **noise points**
  - **overfitting**
- If k is **too large**
  - neighborhood may include **points from other classes**
  - **Underfitting -** influenced by many distant points, not just nearby ones

K-Elbow Plot

Helps to select the optimal number of neighbours for KNN

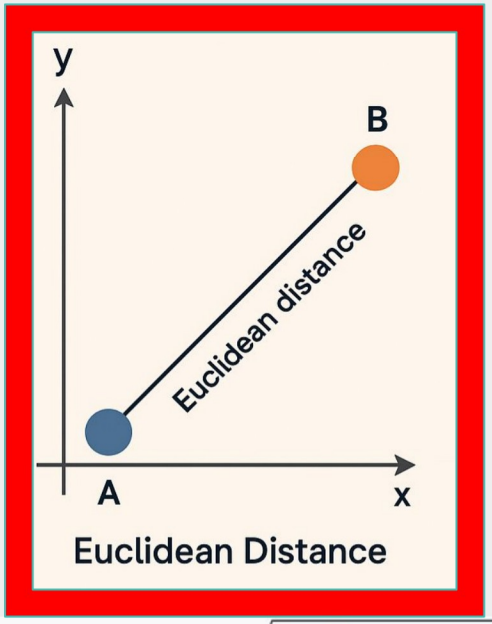


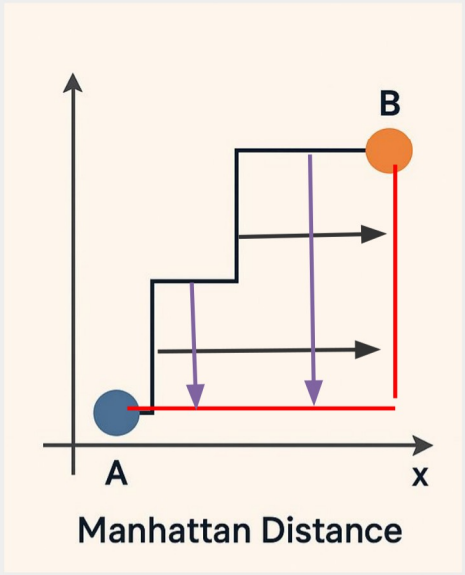
Steps:

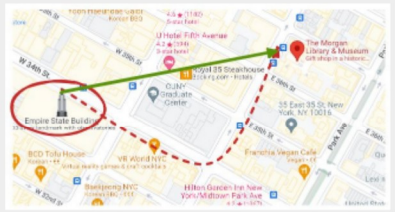
1. Split data into training and validation/test sets.
2. For each k in a range (e.g., 1 to 20):
   - Train KNN with that k
   - Predict on validation set.
3. Compute **error rate** **1 - accuracy rate**
4. Plot k (x-axis) vs error rate (y-axis).
5. Look for the elbow:
   - Point where error **stops decreasing significantly**
6. Choose that k

https://colab.research.google.com/drive/1Lgp-HBmoIMI8CqmzHTbgemNMfTPQ5gg4#scrollTo=11XSWMbVfgdK&forceEdit=true&sandboxMode=true

Distance Measures







$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d}(x_j - X_{i_j})^2]}$$

$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

K-Nearest Neighbours

Pros	Cons
Easy to understand and implement	Slow with large data • Needs to compare every point during prediction.
No training steps • Just stores the data and uses it during prediction.	Curse of dimensionality • Accuracy drops when data has too many features, predictions become less accurate
Few parameters • Only need to set the number of neighbors (k) and a distance method	Can Overfit • Especially when the data is high-dimensional or not clean.

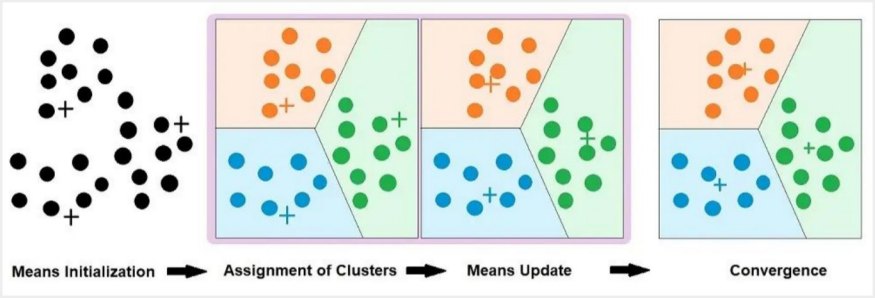
K-Means Clustering

K Means Clustering

- Unsupervised Learning
- Groups **similar data points** into **clusters** without needing labeled data
- Works by grouping points **based on distance** to cluster centers
- “k” → number of groups or clusters to classify items into
- **Key Applications**
  - Data segmentation
  - Anomaly Detection
  - Document Clustering
  - … more

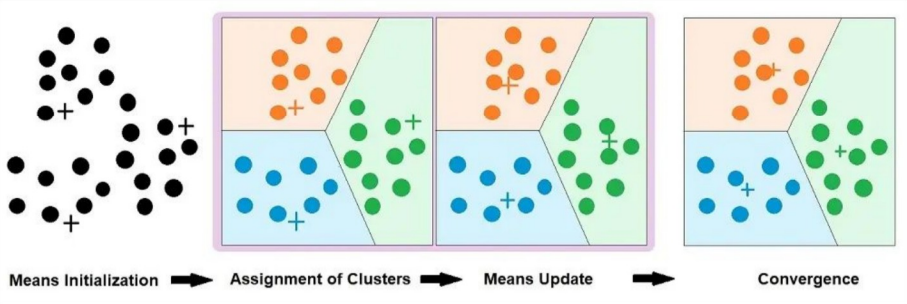
K Means Clustering

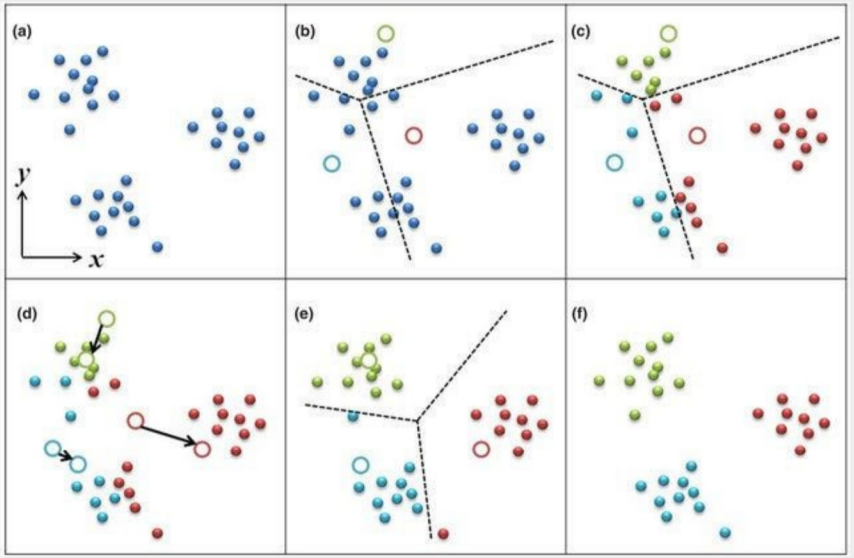
1. Choose **K** (number of clusters)
2. **Initialization**: Randomly selecting **k cluster centroids**.
3. **Assignment Step**
   - Each data point is assigned to the **nearest centroid,** forming **clusters**
     - Find the nearest point using Euclidean distance
4. **Update Step**
   - Calculate **the new centroid** of each cluster
     - Averaging the points within the cluster
5. **Repeat**
   - Goal is to minimise SSE or WSS (Within Cluster Sum of Squares)
6. **Until**
   - **Centroids no longer change**
   - or the maximum number of iterations is reached



https://www.youtube.com/watch?v=5I3Ei69I40s

K Means Clustering



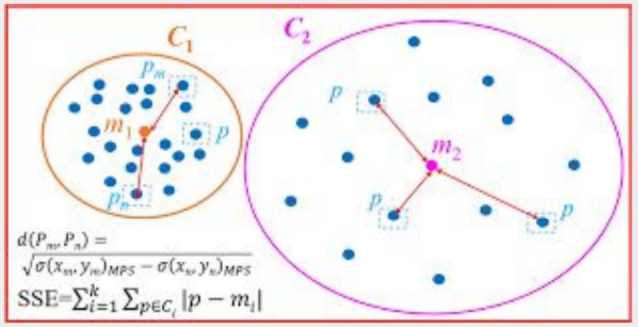


K-Means Clustering - SSE

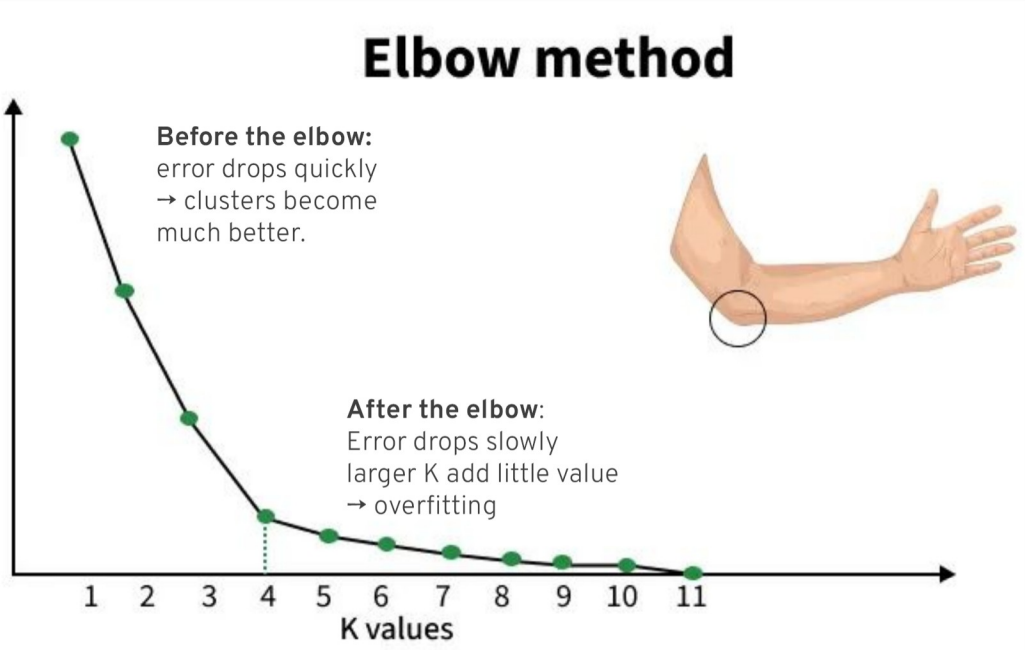
aka Sum of squared errors
aka Within-cluster sum of errors
aka Cluster Inertia

Errors are distances between each observation and its closest centroid

** Note SSE here is different from SSE in Regression*



Selecting the Number of K



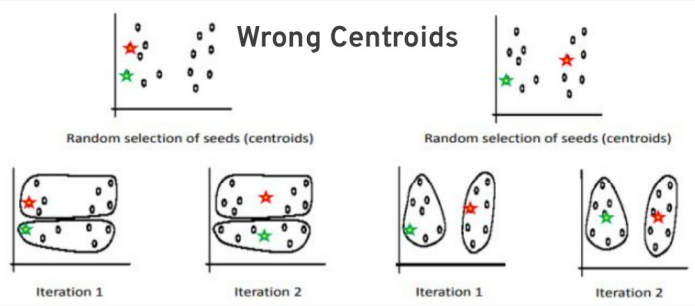
For each k

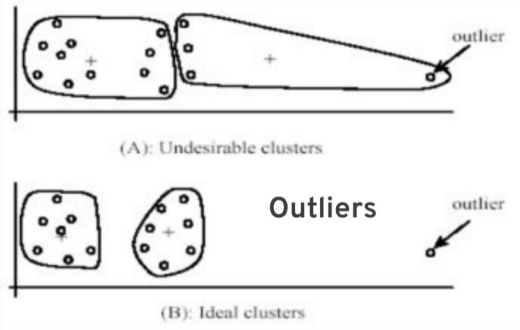
- calculate **SUM of SQUARED ERRORS**
- shows how close the data points are to their cluster centroids

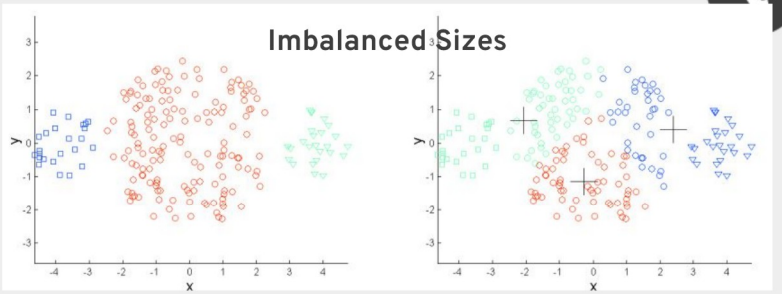
K-Means Clustering

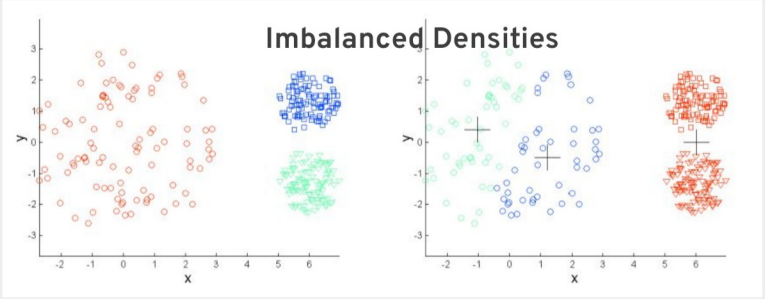
Pros	Cons
Simplicity and Ease of Implementation	Choosing the Right Number of Clusters
Computational Efficiency: K-means is fast and efficient.	Sensitive to Initial Centroids ● The final clusters vary depending on the initial selection of centroids.
Highly scalable: and can efficiently handle large volumes of data points and a large number of variables compared	Non-Spherical Clusters ● Assumes clusters are spherical and equally sized
Easy to interpret and visualize	Sensitive to outliers ● Distort the centroid and, ultimately, the clusters.
	Imbalanced clusters ● Larger clusters dominate centroid updates ● Smaller clusters may be merged or ignored

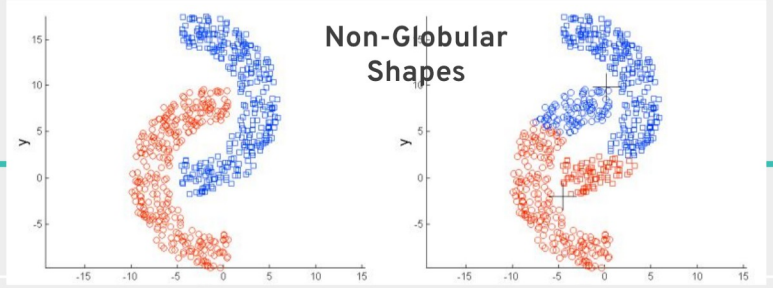
K-Means Clustering











Then, how can you overcome these?

Gaussian Mixture Models

Gaussian Mixture Models

- A **probabilistic model** that represents data as a **weighted combination** of multiple **Gaussian (Normal) distributions**.
- Each Gaussian is a sub-population of the data
- **Soft clustering**: Data points belong to multiple cluster and each point has a probability of belonging to each cluster
  - eg. Observation 23 has
    - 21% chance that it belongs to Cluster 1,
    - 0.1% chance that it belongs to Cluster 2
    - 48% chance of Cluster 3 ...

Gaussian Mixture Models

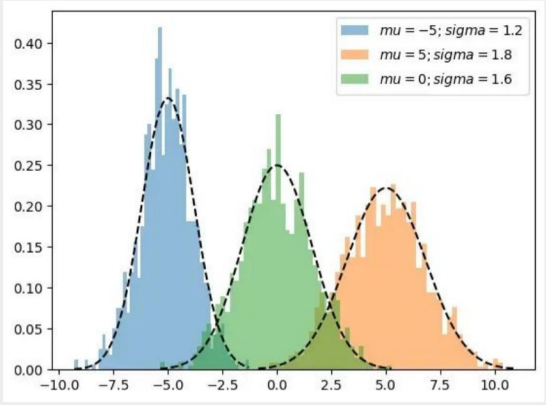
1. **Choose k**: Number of Gaussian components
2. **Initialize parameters**
   a. Mean vectors (μ) *(e.g Centre of cluster)*
   b. Covariance matrices (Σ) *(Spread of Gaussian)*
   c. Mixing weights (π) *(Probability that a data point comes from a particular Gaussian)*
3. **Expectation step (E-step)**: Compute probability each point belongs to each Gaussian

$$P(x) = \sum_{i=1}^{k} \pi_i \mathcal{N}(x \mid \mu_i, \Sigma_i)$$

4. **Maximization step (M-step)**
   a. Update μ, Σ, π to maximize likelihood

$$\log \mathcal{L} = \sum_{j=1}^{n} \log \Big(\sum_{i=1}^{k} \pi_i \mathcal{N}(x_j \mid \mu_i, \Sigma_i) \Big)$$

5. **Repeat**
6. Until log-likelihood **converges**

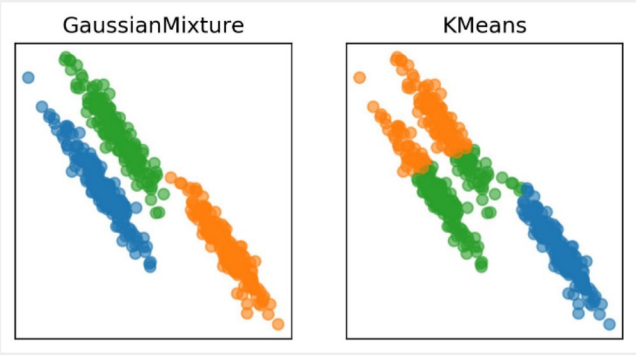


Gaussian Mixture Models

Pros	Cons
Soft Clustering	Sensitivity to Initialization ● Poor initialization can lead to suboptimal solutions or convergence
Clusters of flexible shape and size: GMM offer increased flexibility by allowing users to specify different covariance structures for clusters.	More computationally intensive than K-means ● Estimate cluster means, covariances and mixing coefficients.
Density Estimation: GMM is also a generative model that can estimate the probability density of the underlying data.	Not very scalable ● Large datasets or high-dimensional data may significantly slow down convergence

GMM vs K-Means

	k-means	GMM
Model type	Distance-based	Probabilistic
Cluster assignment	Hard	Soft (probabilities)
Cluster shape	Spherical	Elliptical or variable
Overlapping clusters	Poor	Good
Output	Labels	Labels + probabilities
Metric	Euclidean distance	Likelihood under Gaussian
Speed	Fast	Slower

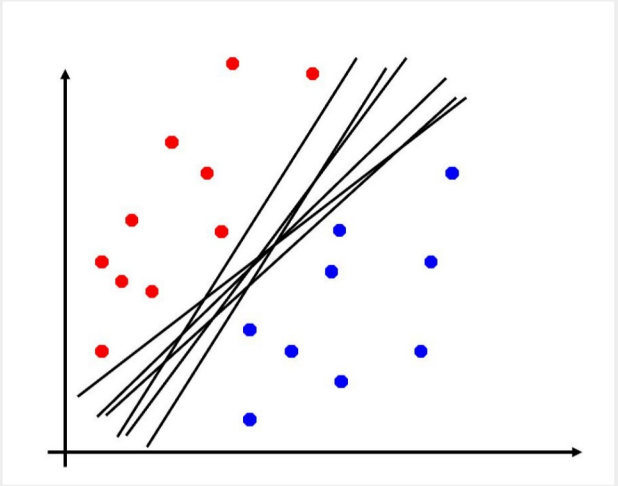


https://colab.research.google.com/drive/1etiuUgT1DxaA7-rcSV-fipyIxW3I8Nu0#forceEdit=true&sandboxMode=true

Support Vector Models

Which Boundary is Best?

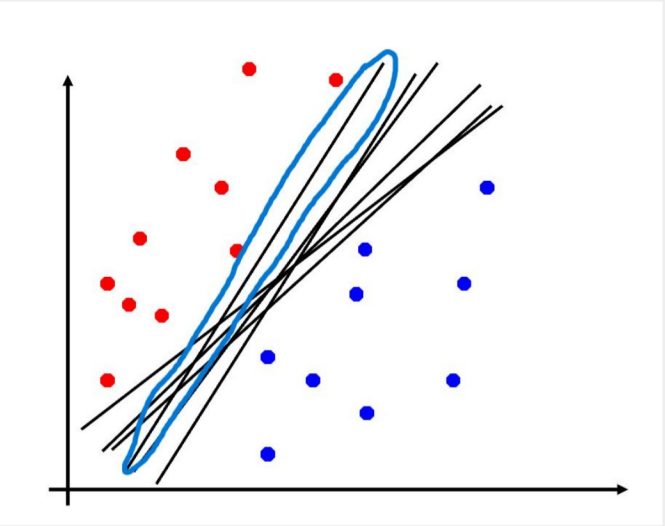
- Multiple lines can separate our data perfectly
- But which one should we choose?
- What happens when new data arrives?



SVM's Answer: Find the boundary that maximizes safety margin

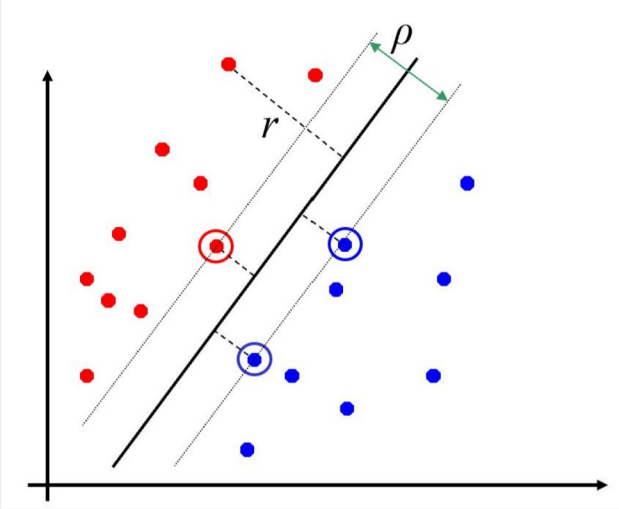
Why The Best Boundary?

- Current boundary very close to the red points
- What if new red point arrives slightly to the right?
- Small shifts in data will cause wrong classifications



SVM: Maximum Margin Classification

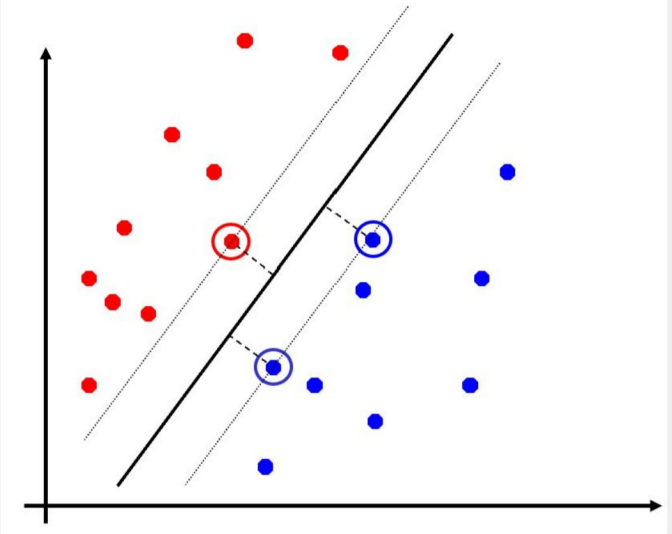
- Margin (ρ) = distance to closest points
- SVM Maximizes this margin
- r is the distance between an arbitrary point to the boundary



- Circled points = support vectors
- Only these points determine the boundary

The Support Vector Concept

- Maximizing the margin is optimal
- Only support vectors matter - remove any other point and the boundary stays identical
- Most real datasets have only a small fraction of points as support vectors



Formulas

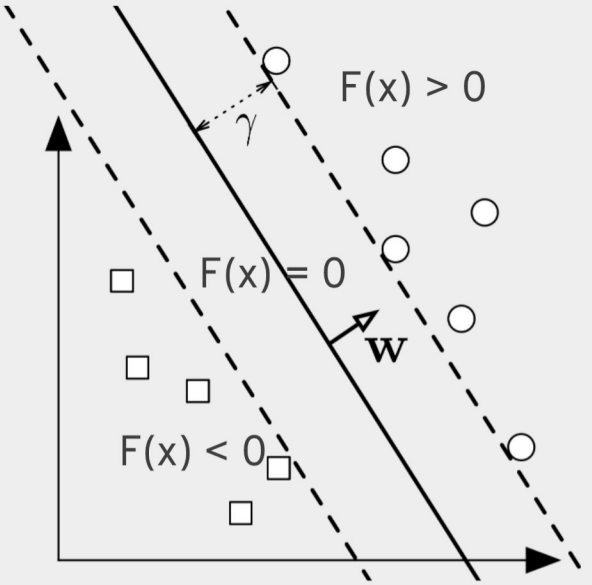
Component	Formula	Meaning
Decision Boundary	$w \cdot x + b = 0$	The hyperplane that separates classes
Classification	$f(x) = \text{sign}(w \cdot x + b)$	Positive side vs negative side
Distance to boundary	$\text{distance} = \lvert w \cdot x + b \rvert / \lVert w \rVert$	Perpendicular distance from any point
Support Vector Constraint	$\lvert w \cdot x + b \rvert = 1$	Support vectors lie on margin boundaries

How SVMs Make Decisions

- SVM decision function: $f(x) = \mathbf{w \cdot x + b = 0}$, sign of f(x) determines class positive or negative

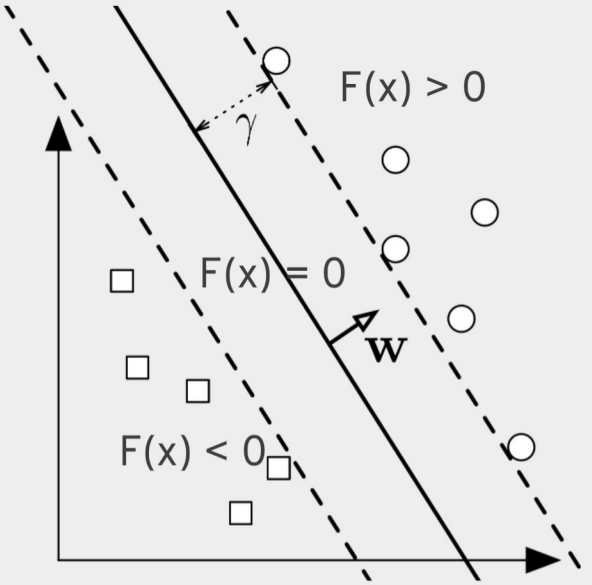
- f(x) > 0 → Class +1
- f(x) < 0 → Class -1
- f(x) = 0 → Exactly on boundary

- Sign follows direction of the normal vector w



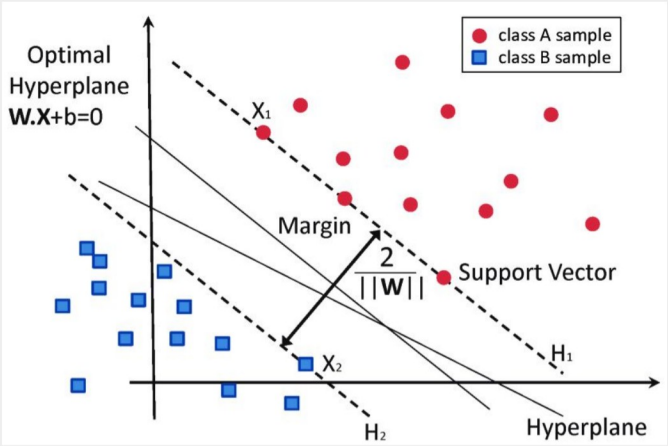
Distance = Confidence

- $|f(x)|$ = distance from hyperplane
- Large $|f(x)|$ → High confidence
- Small $|f(x)|$ → Low confidence



The Optimization Behind SVMs

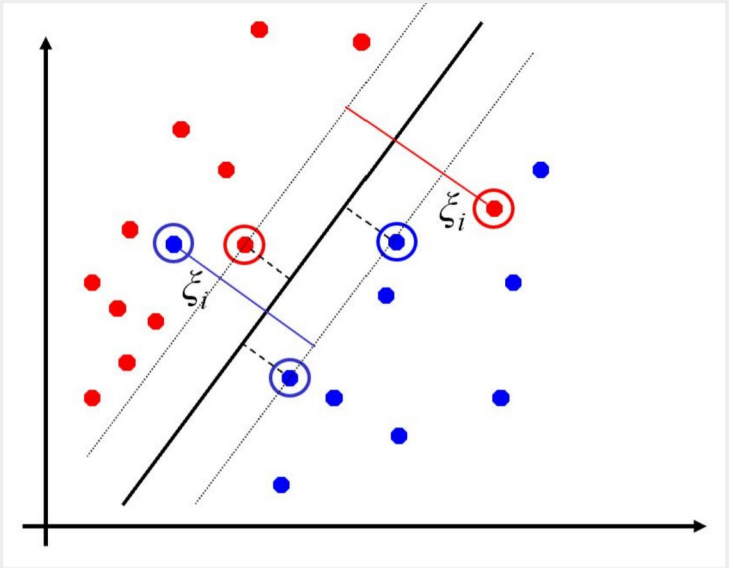
- SVM finds the hyperplane that maximizes the margin



- Mathematical Translation: Minimize $||w||$

- **Wider margin = better generalization**
  - **Margin width = $2/||w||$**
- **To maximize margin → Minimize $||w||$**

Real World Data: Noise



- Perfect linear separation rarely exists in real world datasets
- Soft margin SVMs allow controlled violations of the margin
- ξ_i measures how much a point violates the margin

Slack Variables, ξ_i

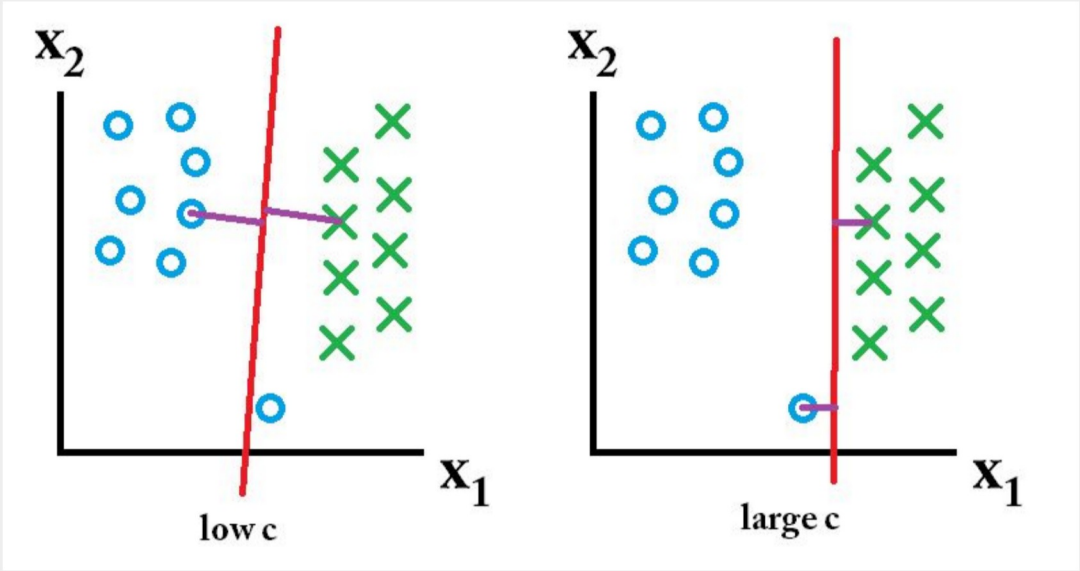
- $\xi_i = 0$
- Point respects the margin
- Either on margin boundary or safely beyond
- No penalty applied

- $0 < \xi_i < 1$
- Point inside the margin but correctly classified
- Violates margin but on the right side of the decision boundary
- Small penalty

- $\xi_i \geq 1$
- Point is misclassified
- On wrong side of decision boundary
- Large penalty

The Role of C

- C Controls the penalty for margin violations in $\|w\|^2 + C\sum\xi_i$.
- High C = Strict enforcement, Low C = lenient enforcement



Controlling the Trade Off

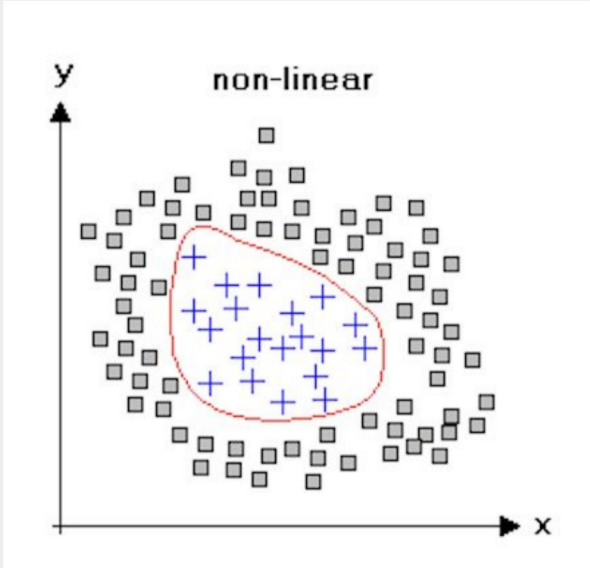
High C (Strict):	Low C (Lenient):
• Diagram shows narrow margin	• Diagram shows wider margin
• Focuses on minimizing errors	• Focuses on maximizing margins
• Risk: Overfitting	• Risk: Underfitting

Lab session!

Let's run through all the SVM fundamentals so far!:

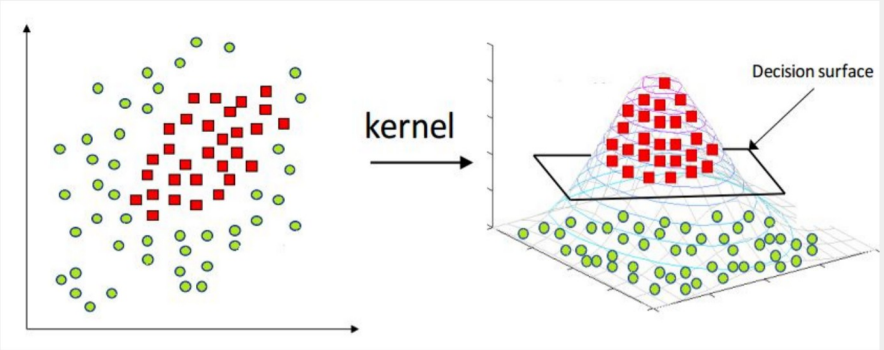
https://claude.ai/public/artifacts/5e81fc26-c146-4264-a682-b014e506cb55

When Linear Boundaries Fail



- Linear SVMs assume data can be separated by straight lines/planes
- What about concentric circles such as this one?
- Needs a curved boundary not a straight line.

The Kernel Trick: Core Idea



- Transform Non-Separable data into a space where linear separation works.
- Compute in high dimensions without going there

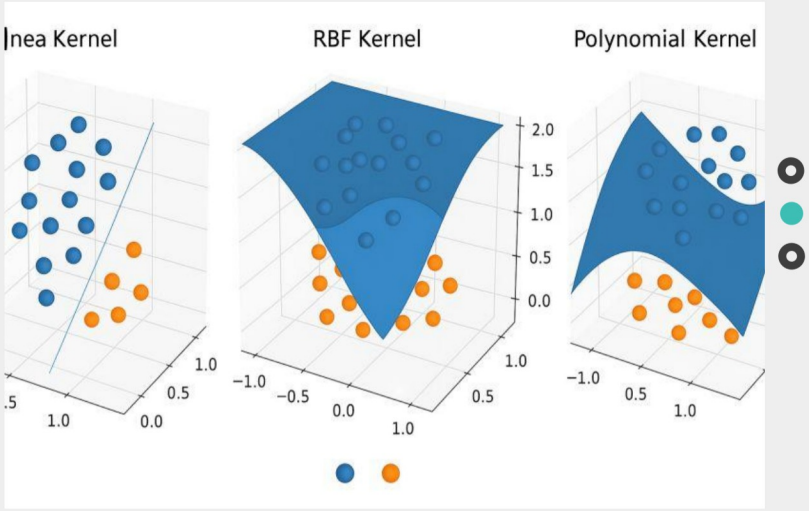
- 2D: No line can separate red from green
- 3D: Clear separation with a flat decision surface
- Magic: Curved boundary appears back in 2D

Essential Kernels

- Linear Kernel: Straight lines and planes
- Used when data is mostly separable

- Polynomial kernel: Curved decision boundaries
- Used when data has polynomial relationships

- RBF: Blob-like decision regions
- Used when data has complex non-linear patterns



Lab session!

Let's run through all kernels now!:

https://claude.ai/public/artifacts/fd92742f-200d-46e3-81b9-b6ab20f80cbf

SVM Strengths & Limitations

Strengths	Limitations
Effective in high-dimensional spaces. SVMs work well even when num of features exceed num samples. Useful for gene expression.	Computationally expensive for large datasets - Training time scales poorly with dataset size.
Memory efficient - only uses support vectors (subset of training data) for decision making.	No probabilistic output - SVMs provide binary classifications without confidence scores, requiring additional techniques like platt scaling.